UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/650,257	08/27/2003	Bhavesh P. Davda	4366-120	7452

48500 7590 04/04/2008

SHERIDAN ROSS P.C.
1560 BROADWAY, SUITE 1200
DENVER, CO 80202

EXAMINER	
WEI, ZHENG	
ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
04/04/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/650,257
Filing Date: August 27, 2003
Appellant(s): DAVDA, BHAVESH P.

Bradley Knepper
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 12/29/2007 appealing from the Office action mailed 04/09/2007

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5619698	Lillich et al.	4-1997
20040107416	Buban et al.	5-1995

20020152455	Hundt et al.	4-2001
20030167463	Munsil et al.	3-2002

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –
(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-5, 7, 10-13 and 16-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Lillich (Lillich et al., US 5,619,698)

Claim 1:
Lillich discloses a method for updating a running process, comprising:

- allocating in executable program code text first memory space operable to receive new program instructions comprising at least a first updated function (See for example, Fig.2, element 130, 132 and related text);
- allocating in executable program code text second memory space operable to receive address information related to said new program instructions (See for example, Fig.2, element 110 and related text, also see col.2, lines19-20, “a trap table resident in RAM”);

- running said executable program code (See for example, col.2, lines 48-50, "During execution of application code 102…");
- stopping execution of said executable program code (See for example, col.2, lines 49-50, "the 68K micro-processor will encounter and execute the ATRAP 104");
- injecting a jump instruction and an address of an update table at a location in a memory containing a first instruction of a first replaced function, wherein said address of said update table contains an address of a first instruction of said first updated function (See for example, col.3, lines 1-5, "the ATRAP instruction 104" (injected jump instruction) and "trap table 110" (conations address to the corresponding system routine) and also Fig.2, Fig.6 teach the art (see for example, Fig.6, element 465 "re-direct code" (jump instruction) and element 469-473, "pointer" (address of update table)));and
- resuming execution of said executable program code, wherein said first updated function is called in place of said first function, and wherein said executable code is updated in said memory (See for example, Fig.2, element 150 and related text, also see col.3, lines 1-5, "and then jumps to the corresponding system routine.")

Claim 2:

Lillich further discloses the method of Claim 1, wherein said step of resuming execution of said executable program code comprises running an intermediate executable, wherein said intermediate executable comprises said updated copy of said executable program code in said memory (See for example, col.3, lines 45-51, "address_1 will point to patch code 132 located in RAM 130 rather than the original system routine").

Claim 3:

Lillich also discloses the method of Claim 1, further comprising: before said step of running said executable program code, copying said executable program code to said memory (See for example, Fig.2, element 122 and related text, "system code").

Claim 4:

Lillich also discloses the method of Claim 1, further comprising:

- injecting a jump instruction and an address of said update table at a location in a stored copy of said executable program code containing an address of said first function (See for example, col.3, lines 1-5, "the trap dispatcher 108"); and
- populating said update table with an address of said first updated function, wherein a stored copy of said executable program code is updated (See for example, Fig.2, element 110, "trap table" and related text).

Claim 5:

Lillich further discloses the method of Claim 4, wherein said updated stored copy of said executable program code comprises final updated executable program code (See for example, Fig.2, elements 146, 150, 152 and related text in col.3, lines 45-51")

Claim 7:

Lillich further discloses the method of Claim 1, wherein said injected jump instruction replaces a first instruction of said first replaced function (See for example, col.3, lines 1-5, "the ATRAP instruction 104").

Claim 10:

Lillich discloses a computer implemented method, the method comprising:

- receiving information identifying: a running executable program to be patched and a function to be replaced (See for example, col.8, line 67-col.9, line 8,"TVector");
- accessing a symbol table in a memory for said executable program to be patched (See for example, col.9, lines 9-13, "binding manager");
- obtaining from said symbol table an address of said function to be replaced (See for example, col.9, lines 10-12, "the address of the code or data");
- stopping execution by a processor f said running executable program to be patched (See for example, fig.14, steps 852 and related text);
- injecting in said running executable program to be patched at a location in said memory containing a first instruction of said function to be replaced a jump instruction and an address of a new function, wherein said new function is executed in place of said function to be replaced, and wherein a patched version of said executable program is created in memory (See for example, col.19, lines 29-40, "the address of the desired extension patch" and also see col.3, lines 1-5, "the ATRAP instruction 104" (injected jump instruction) and "trap table 110" (conations address to the corresponding system routine). The Fig.2 and Fig.6 also teach the art (see for example, Fig.6, element 465 "re-direct code" (jump instruction) and element 469-473, "pointer" (address of update table))) and
- resuming execution of said executable program by said processor, wherein said patched version of said executable program is executed by said processor (See for example, col.19, lines 41-46, "trap dispatcher").

Claim 11:

Lillich discloses the method of Claim 10, further comprising providing a jump table in said memory, wherein said injecting in said running executable program to be patched a jump instruction and an address of a new function comprises replacing a first instruction line associated with said function to be replaced with

an instruction to jump to a first address contained within said jump table, and wherein said first address contained within said jump table comprises an address of a first instruction line of said new function (See for example,Fig.2, element 110 and related text, "trap table").

Claim 12:

Lillich also discloses the method of Claim 11, wherein said first instruction line associated with said function to be replaced comprises a first instruction of said function to be replaced, wherein said first instruction is not an instruction to jump to another address, and wherein said first instruction of said function to be replaced is replaced by said instruction to jump to a first address (See for example, Fig5 and fig.7, element 560, 400 and related text).

Claim 13:

Lillich further discloses the method of Claim 11, wherein said first instruction line associated with said function to be replaced comprises an instruction to jump to a second address contained within said jump table, and wherein said instruction to jump to a second address is replaced by said instruction to jump to a first address. (See for example, Fig5 and fig.7, element 560, 400 and related text, "TVector").

Claim 16:

Lillich also discloses the method of claim 10, wherein said computational component comprises a computer readable storage medium containing instructions for performing the method (See for example, p.5, lines 36-38, "A first aspect of the present invention teaches a computer implemented method for integrating patches into a computer operating system").

Claim 17:
Lillich further discloses the method of claim 10, wherein said computational component comprises a logic circuit. (See for example, Fig.1, element 52, "CPU")

Claims 18-19:
Claims 18 and 19 are system claims for updating executable program code using the methods discussed in claims 10-11. Therefore, accordingly these claims would also be anticipated by Lillich.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 6, 8, 14 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lillich (Lillich et al., US 5,619,698) in view of Buban (Buban et al., US 2004/0107416)

Claim 6:
Lillich discloses the method as in claim 1 above, but dose not further disclose the method further comprising:

- determining a first distance between a position within said code text at which execution of said executable program code is stopped and an address of a first function, wherein said first function is a function to be updated; and
- in response to said first distance exceeding a predetermined amount, populating an update table stored in memory with an address of a first updated function

However, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was make to use this method to determine the distance between those two addresses. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine as once suggested by Buban (See, p.5, paragraph[0046], "atomically updated").

Claim 8:

Lillich and Buban disclose the method as in claim 6 above, but Lillich does not disclose the predetermined amount about the distance. However, Buban further disclose wherein said predetermined amount is 8 bytes. (See, p.5, paragraph[0046], "a 64-bit (8-byte) word on an Intel x86 processor"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was make to use this predetermined amount 8 byte. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine if the processor's smallest unit of atomically replaceable memory is 8 byte (See, p.5, paragraph[0046], "atomically updated").

Claim 14:

Lillich discloses the computational component for performing the method of claim 10, but does not disclose the method further comprising:

- determining a number of bytes between a location within said executable program at which said executable program is stopped and an address of said function to be replaced.

however, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was make to use this method to determine the distance between those two addresses. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

Claim 15:

Lillich and Buban disclose the method of Claim 14, Lillich also discloses in response to said determined number of bytes being at least as great as a first selected number, injecting in a stored copy of said running executable program to be patched said jump instruction and said address of said new function in place of said address of said function to be replaced, wherein a patched version of said executable program is created, but does not disclose the method of injecting is based on selected number. However Buban discloses the reason in response to said determined number of instructions being at least as great as a first selected number (See, p.5, paragraph[0046], "A processor's smallest unit of atomically replaceable memory, e.g., a 64-bit (8-byte) word on an Intel x86 processor"). One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

5. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lillich (Lillich et al., US 5,619,698) in view of Munsil (Munsil et al, US 2003/0167463)

Claim 9:
Lillich discloses the method of Claim 1, but does not disclose wherein said second memory space is read only memory space. However, Munsil discloses a method for applying custom application-compatibility fix using read only memory (see for example, p.4 paragraph[0033], "using read-only memory to prevent modification and corruption by unauthorized or unknowledgeable parities"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was make to use read only memory to store patch address information. One would have been motivated to do so to protect important address information.

6. Claims 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hundt (Hundt et al., US 2002/0152455) in view of Buban (Buban et al., US 2004/0107416)

Claim 20:
Hundt discloses a system for dynamic instrumentation of an executable program, comprising:

- a create patch tool operable to receive information identifying a executable program to be updated and a function to be replaced (See for example, Fig.1, steps 102-108 and related text);
- a patch tool operable to query an operating system for a process identifier associated with said identified executable program (See for example, fig.1, step 104 and related text, also see p.2, paragraph[0030], "an available thread is selected from the application");

- a debugging utility operable to stop execution of said executable program to be updated and to determine a position of an instruction pointer associated with said executable program to be updated (See for example, Fig.1, step 110 and related text. "patch function entry points with breakpoints"); and
- a signal handler tool operable to replace in memory an address of said function to be replaced with an address of a replacement function in response to said position of said instruction pointer being at least a predetermined distance from said address of said replacement function, wherein said replacement function is executed instead of said function to be replaced upon resuming execution of said executable program, wherein said executable program is updated. (See for example, paragraph[0018], "executing the instrumented functions instead of the original functions")

but does not disclose replace memory address according to the predetermined distance. However, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was make to use this method to determine the distance between those two addresses. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

Claim 21:

Hundt and Buban disclose the system of Claim 20 above, Hundt further discloses, the signal handler is additionally operable to replace in a stored copy of said executable program an address of said function to be replaced with an address of a replacement function, wherein said stored copy of said executable program is updated (See for example, Fig.1, step 124 and related text).

(10) Response to Argument

A. Claims 1 and 10 and Dependent Claims 2-9 and 11-17 (Brief pages 10-11):

The Appellant argues that Lillich reference does not disclose a running process is updated by injecting a jump instruction and an address of an updated or new function a location in memory containing a first instruction of a first replaced function or function to be replaced.

In response, said recited limitation about injecting a jump instruction can be reasonable interpreted as inserting/replacing original instruction with a new jump instruction and a new address such that it would direct the microprocessor to execute with the new computer instruction from the new memory address. Therefore, looking at Lillich reference about patching as a whole (Fig.2, all intermediate steps 140, 142, 146, and 148 as a whole), microprocessor executes ATRAP instruction and jumps (jump instruction) to the corresponding system routine indicating by the trap table (memory location) (see for example, col.3, lines 1-5). As Lillich disclosed above, the jump instruction and address of patching code in the memory (trap table) have to be used by the microprocessor to execute the patch code (updated code) instead of execution on old system routine (code being replaced) at location of the first instruction of a first replaced function (system routine). Therefore, Lillich does provide an equivalent function of the cited limitation about injecting a jump instruction and an address of an updated or new function at a location in memory containing a first instruction of a first replaced function or function be replaced.

B. Claims 18-19 (Brief page11):

The Appellant argues that Lillich reference does not disclose recited limitation about stopping execution of code to perform at patch, performing the patch operation and resuming execution of program code in claims 18-19.

In response, as Lillich disclosed at col.2, line 49- col.3, line 5, "the microprocessor pushes some state onto the computer system stack, the state including the address of a ATRAP instruction…", and Fig.2 clearly indicates that Application code 102 stopping execution at ATRAP 104, microprocessor using stack to save states for resuming execution, then performing/directing the patch code operation step 150 and resuming execution of application 102 by step 148 (see for example, col.2, lines 48-col.3, line 5, also see col.3, lines 39-51). Therefore, Lillich does disclose the cited limitations as in claims 18-19.

C. Claims 6, 8 14 and 15 (Brief page 13):

The Appellant argues that there is no mention in the cited portions of Buban of determining a distance between a position within code where execution is stopped and address of a function be updated or replaced (emphasis added).

In response, as Buban reference disclosed in paragraph 47, in order to "To find a suitable instruction to replace (where), it may be necessary to move backwards in the existing code (where)…and then add instructions (replace) to the beginning of the patched code or to move forward and then add instructions to the front of the patched instructions (emphasis added). Therefore, such determining actions including moving

backwards/forwards and, where for, adding instructions have to determine the distance between codes so that it will further direct the instruction to execution in the memory properly, otherwise, it would be inoperative. Therefore Buban does disclose said limitation as claimed.

D. Claim 9 (Brief page 13):

The Appellant merely relies upon that Munsil does not make up of the deficiencies of the Lillich reference with respect to independent Claim 1.

As addressed in (A) above, Lillich does indeed teach such claimed limitation as argued, accordingly Appellant's argument for claim 9 is also not persuasive. Furthermore, Munsil discloses the purpose of using ROM to prevent of unauthorized or unknowledgeable modification (see for example, paragraph 0033]). Therefore, Lillich and Munsil together do disclose all the limitations of claim 9.

F. Claims 20 and 21 (Brief page 14):

The Appellant argues that Hundt reference and Buban reference do not teach replacing in memory an address of function to be replaced with an address of a replacement function in response to the instruction pointer being at least a predetermined distance from the address of the replacement function.

As addressed in (C) above, Buban reference disclosed in paragraph 47, in order to "To find a suitable instruction to replace (where), it may be necessary to move backwards in the existing code (where)…and then add instructions (replace) to the

beginning of the patched code or to move forward and then add instructions to the front of the patched instructions" (emphasis added). Therefore, such determining actions including moving backwards/forwards and, where for, adding instructions have to determine the distance between codes so that it will further direct the instruction to execution in the memory properly, otherwise, it would be inoperative. Therefore Buban does disclose said limitation as claimed.

It should be noted the Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Zheng Wei/

Tuesday, March 25, 2008

Conferees:

SPE Tuan Q. Dam, Group Art Unit 2192

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192

SPE Eddie C. Lee, Technology Center 2100

/Eddie C Lee/

Supervisory Patent Examiner, TC 2100